Technische Universität Ilmenau
Fakultät für Mathematik
und Naturwissenschaften
Institut für Mathematik

Postfach 10 0565
98684 Ilmenau
Germany
Tel.: 03677/693267
Fax: 03677/691241
Telex: 33 84 23 tuil d.
email: W.Neundorf@mathematik.tu-ilmenau.de

Preprint No. M 4/96

3D-Darstellung von Funktionen

Thomas Lutter, Werner Neundorf

März 1996

‡MSC (1991): 65-01, 65Y99, 68U99, 92J99

Die erfolgreiche Anwendung von Lösungsverfahren hängt nicht nur von ausgereiften Softwaretools ab, sondern in der ersten Phase der Lösung ist eine breite fachliche Grundbildung für Problemanalyse, für Möglichkeiten der Umformung einer Aufgabe in eine andere adäquate Darstellung sowie für die Auswahl von entsprechenden Lösungsmethoden notwendig. Dieser Kenntnisstand sowie die Berücksichtigung der Rechnerarchitektur und Eigenschaften des Computers kann zu einer höheren Qualität an Verfahren und darauf aufbauenden Computerprogrammen führen. Andererseits verfügen auch Softwaretools über zahlreiche tutorielle Elemente oder Bausteine, und sie können in vielfältiger Weise zu tiefergreifenden Überlegungen und neuen Ideen führen.
Dabei erweisen sich überschaubare Programme als wirksame und moderne didaktische Werkzeuge. Sie gestatten insbesondere die weitgehende Konfigurierbarkeit und damit kognitive Manipulationsmöglichkeiten, auch bezüglich graphischer und algebraisch-symbolischer Elemente, Darstellungsweisen und Umfang der Ergebnisse, Funktions- und Methodenauswahl, interaktive Arbeitsweise und Wiederholbarkeit.
Ausgewählte Beispiele und Demonstrationen aus dem Gebiet der räumlichen Darstellungen von Funktionen sollen dies hier untersetzen.

Successful application of numerical solvers depends not only on fully-developed software tools, but during the primary phase of solution there is necessary a wide basic education which includes the ability for problem modelling and analysis, possibilities of transforming the problem into another form and the choice of appropriate solution methods. This knowledge level and the consideration of machine architectures and computer features can lead to higher quality of methods and computer programs, which are based on the algorithms. On the other hand, software tools, too, dispose of tutorial components and they can promote deeper reflections and new powerful ideas in varied ways.
Therefore, limited programs prove as effective and modern didactic means. They allow especially the far-reaching changes of configuration and in this way some possibilities of doing cognitive manipulations with regard to graphical and algebraic-symbolic elements, presentation and scope of results, selection of functions and methods, interactive mode of operation and repetition.
This is supported in the paper by selected examples and demonstrations from the area of spatial presentation of functions.

Key words: tutorial aspects, programs, computer graphics, computer aided design.

MSC (1991): 65-01, 65-04, 65D17, 65Y25, 68-04, 68U05

1 Einleitung

Beobachtungen und Erfahrungen zeigen, daß es für Studenten der ersten Semester in naturwissenschaftlichen und technischen Fachrichtungen nicht unbedingt notwendig ist, das große Szenarium eines noch größeren Tools nahezubringen. Vielmehr erweisen sich überschaubare Programme als wirksame und moderne didaktische Werkzeuge. Sie gestatten insbesondere die weitgehende Konfigurierbarkeit und damit kognitive Manipulationsmöglichkeiten, auch bezüglich graphischer und algebraisch-symbolischer Elemente, Darstellungsweisen und Umfang der Ergebnisse, Funktions- und Methodenauswahl, interaktive Arbeitsweise und Wiederholbarkeit (vgl. [1]).

Auswahl einiger Softwareaspekte als methodisch-didaktische Hilfsmittel in kognitiven Modellen :

- Multiple-window Darstellungen, Desk-Top-Publishing Systeme, Menütechnik,
- Computergraphik, Animation,
- Nutzung von Gleichungen und Funktionsgraphen,
- algebraische, geometrische und graphische Symbolsysteme,
- Darstellung von Formeln und Tabellen,
- Datenspeicherung und -manipulation,
- Wiederholbarkeit, Reflexion, Interaktion, Verstärkeraspekt, Reorganisationsaspekt.

An Bedeutung gewinnen somit neben den numerischen Aufgabenteilen auch die geschickte Umsetzung einer Anwendungssituation in ein passendes Softwarekonzept mit unterschiedlichen Darstellungsmöglichkeiten, die auch weitere Entwicklung und Verbesserungen gestattet.

Dies soll nunmehr an einigen ausgewählten Beispielen aus dem Komplex der räumlichen Darstellung von Funktionen mit Softwarelösungen in Pascal demonstriert werden. Dazu werden weiterhin einige Details der Programmierung angegeben. Seitens der numerischen Probleme und Verfahren sowie der Informatik sind dabei zu berücksichtigen z.B.:

- Lösung von Gleichungssystemen, Matrixkondition,
- Termumformungen, mathematische und numerische Äquivalenz,
- Fehleranalyse,
- Interpolation auf der Basis von Funktionen bzw. Referenzen, Kurvendiskussion,
- Verarbeitung von umfangreichen Datenmengen, Datensicherung, kontrollierte E/A
- 3D-Graphik (vgl. [9],[10],[11]),
- Dialogaufbau, Entwicklungsumgebungen und Oberflächen,
- spezielle Computereffekte.

Die Beispiele weisen u.a. darauf hin, wie komplex die Verwendung des Computers als Werkzeug und Medium der Lernens und Lehrens allgemein und besonders in der Ausbildung von Studierenden zu sehen ist.

2 3D-Plot von Funktionen

2.1 Programm PLOT3D32.PAS

- (C) **W.Neundorf IfMath TUIlmenau 1995**
- *Programmiersprache* : **Turbo Pascal V6.0,7.0, Borland Pascal V7.0**
- *Aufruf* : **PLOT3D32.EXE**
- *Modul* : **Unit FOR_COM5.PAS für Funktionsparser** $f(x, y, p)$
- *Dokumentation* : **Quelltext, selbstdokumentierend**
- *Hardwarevoraussetzung* : **mind. 386er PC, VGA-Graphikkarte, Koprozessor**
- **Shareware**
- **Inhalt**
  Plot der reellen Funktion $z = f(x, y, p)$, $p[1..n]$ Parametervektor, im rechteckigen Bereich von $\mathbb{R}^2$

2.1.1 Kurzcharakteristik

- **Menüorientiert, wahlweise Ablaufsteuerung, benutzerfreundliche Oberfläche, Fenstertechnik (4 kleine oder 1 großes Fenster)**
- **Auswahl von Funktionen** $f(x, y, p)$ **(Anzahl=14) oder Funktionseingabe sowie Formelcompiler (Funktionsparser)**
- **Dateiarbeit, Datei für** (x, y, z)**, vorhandene Beispieldateien**
- z**-Streckung der Funktion, Ansicht gemäß Augpunkt (Winkel), Ausdünnen des Punktgitters für Darstellung (fein, normal, grob)**
- **Demonstrationsvariante**
- **Animation des Plots**
- **Druck von auswählbaren Bereichen des Plots (24-Nadel-Drucker)**

2.1.2 Informationen und Menüs aus dem Programm

- 3D-Darstellung der Funktion $z = f(x, y, p)$
  Für ein rechteckiges Gebiet der Argumente (x, y) kann eine ausgewählte Funktion dargestellt und ihre Betrachtung manipuliert (drehen, strecken, verfeinern, drucken) werden. Die 2D-Simulation von $z = f(x, y, p) = f(x, p)$ ist durch die Wahl eine sehr schmalen y-Intervalls möglich. Die übliche Vorgehensweise ist :
  1. Startmenü

```
     S T A R T M E N U E

  ▷ about - zu beginn genau lesen !
    funktion Definieren
    file Erstellen
    Automatischer ablauf mit protokoll
    infos zu Gitter fuer graphik und augpunkt
    Naechstes menue

    Halt
```

2. Auswahl einer Funktion oder Eingabe dieser als String mit Formelcompiler
3. Datenaufbereitung und File erstellen
   j,n; x[1],y[1],z; x[1],y[2],z; x[1],y[3],z; .... ;x[1],y[n],z;
   x[2],y[1],z; x[2],y[2],z; x[2],y[3],z; .... ;x[2],y[n],z;
   ...
   x[j],y[1],z; x[j],y[2],z; x[j],y[3],z; .... ;x[j],y[n],z
   alle Werte als SINGLE abgelegt
   Eingabe: $2 \leq$ j, Anfangswert x und Schrittweite xh
   $2 \leq$ n, Anfangswert y und Schrittweite yh , j*n$\leq$4000
   d.h. x,y bilden äquidistantes Rechteckgitter
4. Nächstes Menü, Laden der Datei,
   Auswahl der gewünschten Datendichte (Gitter grob=2, norm=1, fein=0)
   und Streckungsparameter (fw Wichten=Strecken) für die 3D-Darstellung
5. Darstellung und Manipulieren in 1 oder 4 Fenstern des Bildschirms,
   u.a. Drehen um Achsen, Winkel des Augpunktes, Automatik des Drehens
6. Zurück und Drucken von ausgewählten BS-Bereichen (Print),
   dabei wird das Funktionsgraph auf dem großen BS angezeigt und
   nach einigen Druckereinstellungen und den BS-Bereich (Rechteck) gefragt,
   (Zahleneingabe mit <ET> abschließen, Abfrage wird nicht gedruckt)

```
So drucken? (J/N) :
-------------------------------------------------
D-Modus? (vert=0, horiz=1...4,6): ...
BS-Bereich (lox,loy)-->(rux,ruy): ... ... ... ...
Oberer Rand  (0,1,...Leerzeilen): ...
Linker Rand (0,1,...Leerspalten): ...

Esc --> Abbruch der Druckerausgabe
```

Wenn man sich die Filearbeit ersparen will, läuft eine Demo fast automatisch mit einem temporären Datenfile FXY3DD.DAT, daß jedoch im aktuellen Verzeichnis eingetragen wird. Dieses Datenfile ist zu einem späteren Zeitpunkt gegebenenfalls zu löschen. Keine besondere Datensicherung bei Eingaben.

- Datenauswertung für 3D-Demo
  Wenn hinreichend viele Daten (x, y) zur Verfügung stehen (großes enges äquidistantes Gitter), kann durch Auswahl die graphische Ausgabe in den Fenstern in 3 Varianten erfolgen. Bei weniger Daten erfolgt keine „Ausdünnung“ des Punktgitters.

Punkte	großes Fenster	kleines Fenster(1/4)
Grob (2)	30 x 30	15 x 15
Norm (1)	60 x 60	30 x 30
Fein (0)	120 x 120	60 x 60

Tab.1. Gitterstruktur für großes und kleines Anzeigefenster

Blick von Winkel	oben z=∞	unten z=-∞	rechts y=∞	links y=-∞	vorne x=∞	hinten x=-∞	Punkt (9,9,9)
$xw°$	0	0	90	90	90	90	225
$yw°$	180	0	180	0	90	270	85
$zw°$	180	0	0	0	0	0	135

Tab.2. Einstellung der Drehwinkel für ausgewählte Betrachtungen
- Augpunkte -

- Menübildschirme

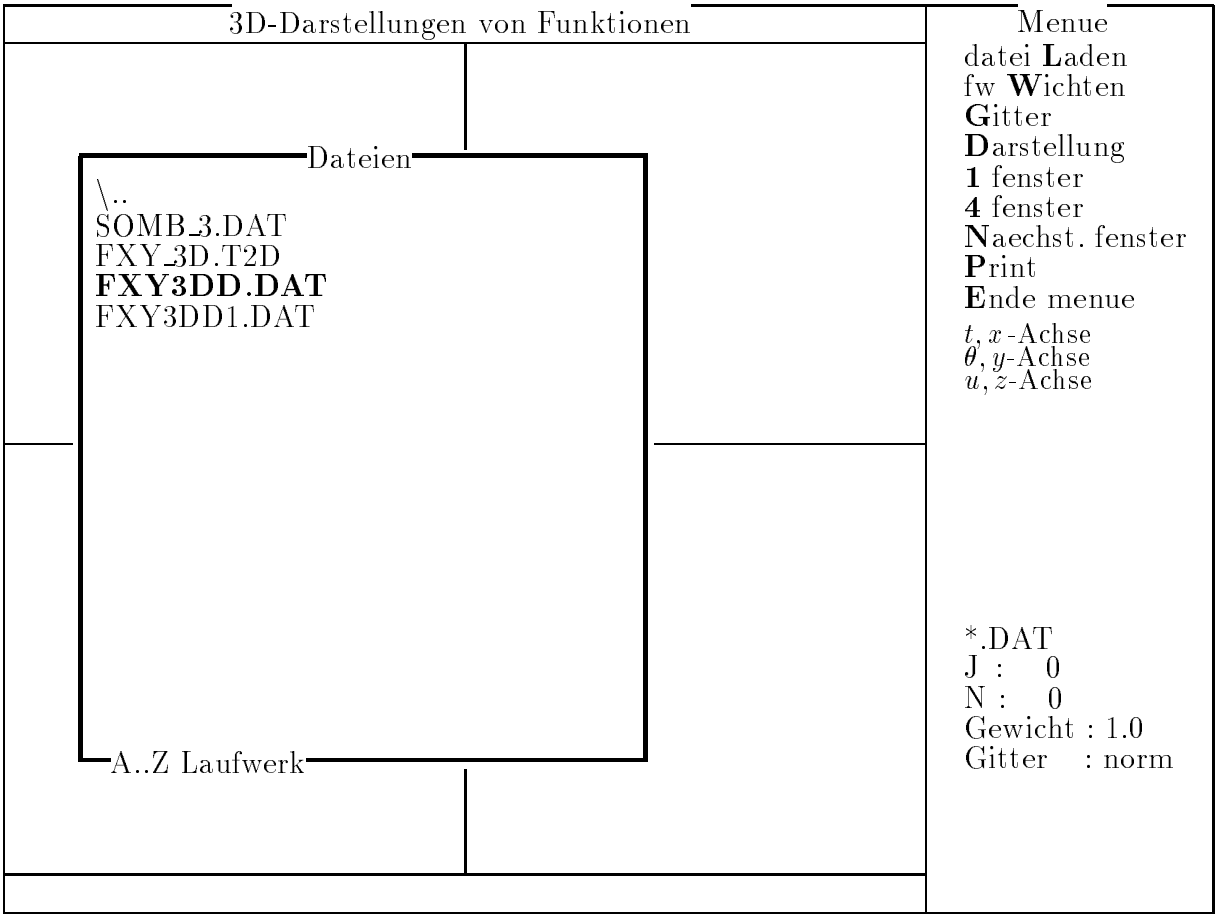


Abb.1. Hauptmenü mit Aufruf des Menüpunktes „datei **L**aden“

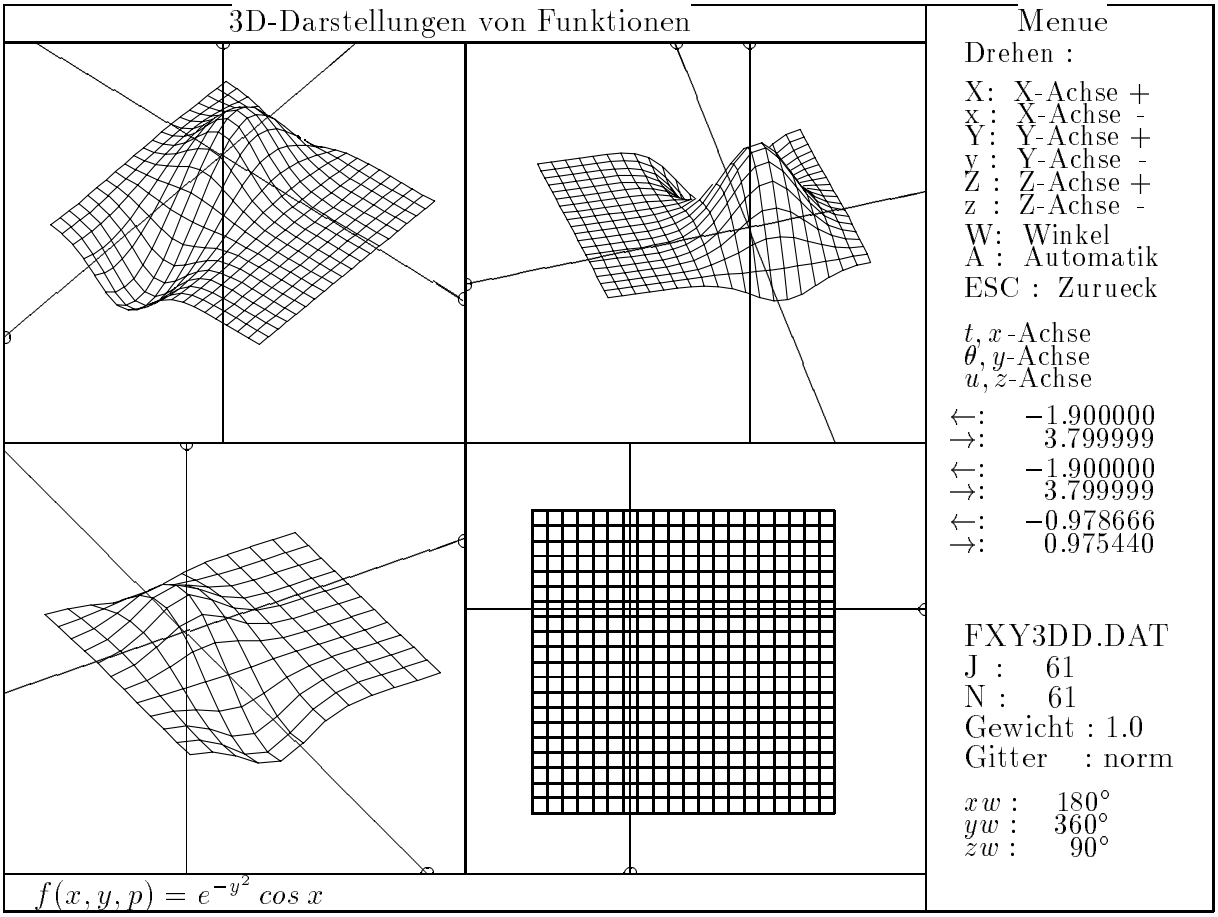


Abb.2. Menü „Drehen“ mit verschiedenen Ansichten eines Funktionsgraphen

- Funktionswahl, Variante 0..14

```
0 = ZK-Eingabe der Funktion f(x,y,p)
1                    sin(x)/x falls x<>0, sonst 1
x*x                  1/(x+1) falls x<>-1, sonst 1E37
x*x*x                -y/(9+sqr(x)+sqr(y))
exp(-x*x)            abs(sin(x))+cos(5*y)
sin(2*x)             20*cos(sqr(x)/4+sqr(y)/4)/(sqr(x)+sqr(y)+3)
cos(x)+cos(y)        -cos(y*pi/10)*cos(3*x*pi/10)
cos(x)+sin(y)        x*x-2*y*y
-------------------------------------------------------------------
Syntax: f(x,y,p) = <expression>  <ET>
        p Parametersatz (Vektor p[1..indexmax])

Nutzbare arithmetische Standard- und Programmfunktionen
(als Quelltext definiert), Konstanten sowie Operatoren
-------------------------------------------------------------------
Sqr     Sqrt    Sin     Cos     Tan     ArcTan
Ln      Exp     Abs     Int     Frac    Round
Trunc   Succ    Pred    Sinh    Cosh    ArTanh
Hi      Lo      Swap    Ord     Random
Random  WhereX  WhereY  Pi
+ - * / ^(Potenzierung, Prioritaet wie * ) -(Vorzeichen)
runde Klammern ()   Indexklammern []   Spaces beliebig
```

- Funktionsplot im Menü „Drehen“
  Hier wurden die TEX-Möglichkeiten von GNUPLOT angewendet (vgl. [8]). Zunächst wird mit dem TP-Programm eine Textdatei TBS11.DAT der Gitterfunktion bestehend aus den Spalten $(x, y, f(x, y))$ erstellt.

```
var datei : text;
    ...
assign(datei,'TBS11.DAT');
rewrite(datei);
x:=xa;
for i:=1 to j do
  begin
    y:=ya;
    for k:=1 to n do
      begin
        z:=fz(x,y);
        writeln(datei,x:6:3,' ',y:6:3,' ',z:6:3);
        y:=y+ys
      end;
    x:=x+xs
  end;
close(datei);
```

  Dieses Datenfile mit der Folge der Tripel (x, y, z) kann man in GNUPLOT graphisch darstellen und in analoger Weise auch als TEX-File TBS11.TEX aufbereiten, um es wie hier in einen Text (Artikel) als Input-File einzubinden. Wir notieren dazu die GNUPLOT-Kommandos, die als File TBS11.PLT zusammengefaßt sind.

```
# TBS11.PLT
set terminal emtex
set output "tbs11.tex"
set size 0.5/1.0, 4.0/5.0
set nokey
set noborder
set clip
set surface
set dgrid3d 20, 20, 2
set nogrid
set hidden3d
set view 30, 130, 1.2, 1
set nozeroaxis
set noxtics
set noytics
set noztics
f(x,y) = exp(-y*y)*cos(x)
set parametric
splot "tbs11.dat" using 1:2:3 with lines 1
pause -1 " EmTex-File erzeugt"
```

Der Plot erfolgt ohne Achsen, Maßstäbe , Beschriftungen usw. Er berücksichtigt den Datenumfang. Natürlich sollte man zuvor sich den Graph im GNUPLOT-Fenster anschauen (*set terminal windows*).
Die Dateiausgabe mit *set terminal latex* erzeugt ein TEX-File, das ca. doppelt so groß ist. Möglich ist auch das Plot-Kommando

splot [-2 : 4] [-2 : 4] [-1.0 : 1.1] *'tbs11.dat' with lines 1*

welches eine leicht abgewandelte Graphik erzeugt, die noch zu skalieren ist (z.B. *set view* 30, 130, 0.5, 1).

- Programmkomponenten
  Für verschiedene Aufgabenstellungen ist es sehr nützlich, sich eine kleine Bibliothek von Unterprogrammen und Moduln bereitzustellen, die auch hier Anwendung finden.
  - Aktivitätsanzeige in Schleifen und anderen Strukturen als rotierender Strich, auch auf dem Graphik-BS wirksam

```
var kreiselvar : byte;
procedure kreisel;
var c : char;
begin {-------------------------------------------kreisel}
  kreiselvar:=1+(kreiselvar mod 16);
  case kreiselvar of
    1..4  : c:='-';
    5..8  : c:='\';
    9..12 : c:='|';
   13..16 : c:='/'
  end;
  gotoxy(79,25);
  write(c)
end; {-------------------------------------------kreisel}
```

  - Kontrollierte und verdeckte Eingabe von Integer-Größen mit <ET>

```
procedure in_integer(var i_stri : string; var i_number : integer);
          { i_stri    - Zahl als Zeichenkette
            i_number  - gueltige Zahl         }
const cr = #13;
var   hstr   : string;
      i,code : integer;
      ch     : char;
begin {-------------------------------------in_integer}
  repeat
    hstr:='';
    repeat
      ch:=readkey;
      hstr:=hstr+ch
    until ch=cr;
    while pos(' ',htr2)>0 do delete(hstr,pos(' ',hstr),1);
    delete(hstr,length(hstr),1);
    val(hstr,i,code)
  until code=0;
  i_stri:=hstr;
  i_number:=i
end; {-------------------------------------in_integer}
```

- Cursor im Text-Bildschirm ein/aus mittels Software-Interrupt $10=10h=16

```
procedure SetCursor(Size : byte);
 { Size = 0   Clear cursor
          1   Set standard cursor  }
var R : Registers;
    PastMode : word;
begin
  PastMode:=LastMode; {momentan aktiver bzw. aktueller Videomodus}
  R.AX:=$0100;
  with R do
  if Size=0 then CX:=$2000  {Cursorgestalt}
            else
    if PastMode=Mono then CX:=$0C0D
                     else CX:=$0607;
  Intr($10,R)
end;
```

- Kontrolle auf Bereitschaft des 24-Nadel-Drucker bzw. Tintenstrahldruckers mittels Software-Interrupt $17=17h=23

```
funktion drucker_ok : boolean;
var R : Registers;
begin
  R.AH:=2;
  R.DX:=0;
  Intr($17,R);
  drucker_ok:=((R.AH and 128)<>0) and ((R.AH and  32) =0) and
              ((R.AH and   1) =0) and ((R.AH and   8) =0) and
              ((R.AH and  16)<>0)
end;
```

2.2 Programm 3D_INT.PAS

- (C) Th.Lutter, W.Neundorf IfMath TUIlmenau 1996
- *Programmiersprache* : Turbo Pascal V6.0,7.0, Borland Pascal V7.0
- *Aufruf* : 3D_INT.EXE
- *Hilfsprogramm* : DATEI_ER.PAS
  Erzeugung der Datei *balken.dat* der Menüleiste sowie
  von Dateien *menue0.dat* ... *menue5.dat* der Menütafeln
- *Modul* : Unit FOR_COM5.PAS Funktionsparser $f(x, y, p)$,
  Parametervektor p wird nicht gebraucht
  Unit MAUS.PAS Graphik-Maus
  Unit LINGLEI.PAS Gleichungsystemlöser (VGA)
  Unit INTER3.PAS Berechnung des Interpolationspolynoms
  $p_n(x, y) = c_1 + c_2x + c_3y + ...$
  Unit BS_AUS.PAS Druckerausgabe eines BS-Fensters
  (24-Nadel-Drucker, Tintenstrahldrucker)
- *Dokumentation* : Quelltext, selbstdokumentierend
- *Hardwarevoraussetzung* : mind. 386er PC, VGA-Graphikkarte, Koprozessor
- Shareware
- Inhalt
  Plot und Interpolation der reellen Funktion $z = f(x, y)$ bzw. auswählbarer Formfunktionen im rechteckigen oder dreieckigen Bereich von $\mathbb{R}^2$

2.2.1 Kurzcharakteristik

- Menüorientiert, Maus, wahlweise Ablaufsteuerung, benutzerfreundliche Oberfläche, Fenstertechnik (Menü- und Statuszeile, Anzeige-, Schalter- und Infofenster),
- Auswahl von $f(x, y)$ durch Funktionseingabe bzw. durch Nummer der Formfunktion sowie Funktionsparser
- Menüdateien mittels DATEI_ER.PAS erzeugen und verändern
- Interpolation mittels 2D-Polynome auf Rechteck, unterteiltem Rechteck (max. 4*4 Elemente) bzw. Dreieck (wird eingebettet in Rechteck) mit Genauigkeit 2,3,4 bei entsprechender Stützstellenverteilung und Funktionswertvorgabe
- Festlegung von Formfunktionen und Genauigkeiten im Quelltext möglich
- Funktion, Interpolationspolynom bzw. Fehlerfunktion als Punkte, Drahtgitter oder farbiges Gebirge
- Ansicht und Bewegen des Plots auf Viereckgebiet
- Bei Dreieckgebiet abschneiden des Plots auf dieses
- Druck des angezeigten Funktionsgraphen mit Rahmen und Titel

2.2.2 Informationen und Menüs aus dem Programm

- Programmbedienung
  Nach Programmstart erscheint kurz ein Eröffnungsbild.
  Danach testet das Programm, ob die erforderlichen Startdateien vorhanden sind. Falls kein Fehler vorliegt, erscheint das Hauptmenü, dazu am oberen Bildschirmrand eine Menüleiste und am unteren eine Statuszeile. Letztere enthält Informationen über weitere Optionen der Programmbedienung oder Fehlermeldungen, die während der Programmabarbeitung aufgetreten sind.
  Die in der Menüleiste angegebenen Menüs können über Mausklick auf den entsprechenden Namen oder das Drücken des rot markierten Buchstabens aufgerufen werden. Das Verlassen der Menues geschieht über <ENTER> bzw. einen Mausklick auf den ausgewählten Menüpunkt oder <ESC> bzw. dem Aufruf eines anderen Menüs, wobei <ESC> bzw. das Aufrufen eines anderen Menüs ein Verlassen ohne Veränderung der bisherigen Einstellungen bzw. ohne Programmabarbeitung zur Folge hat. <ENTER> bzw. einen Mausklick auf den ausgewählten Menüpunkt führt hingegen zu einer Aktualisierung der Einstellungen bzw. einer Programmabarbeitung.

Objekt	Initialisierung
Gebiet=Rechteck	$intart = true$
Viereckskoordinaten	$[xv, xn] * [yv, yn] = [-1, 1] * [-1, 1]$
Funktion $f(x, y)$	0
1 Element (keine Rechteckunterteilung)	$bereichx = bereichy = 1$, max. 4
Genauigkeit = Anzahl der äquid.	
Randknoten je Seite	$genau = 2$, max. 4
= Zahl der Knoten auf Seiten des lok. V.	$ixa = ixb = iya = iyb = 2$, max. 4
Perspektive	$perspektive = 0$
Schrittweite der Perspektive	$dpers = 0.01$
Maximale Perspektive	$= mpers + dpers = 0.20$
Drehwinkel	$wx = wy = wz = 0$
Drehwinkelschrittweite	$drehw = \pi/30$
Gitterauflösung	$idx = idy = 24$, max. 25*25
Funktion $f(x, y)$ nicht darstellen	$showfkt = false$
Funktion p_n, $p_n - f$ als Punkte darstellen	$ansicht = 1$ (bei Ansicht)
	$darst = 1$ (bei Bewegung)
Fehlerfunktion $p_n - f$ nicht anzeigen	$differenz = false$
Bild nicht drucken	$print\_ok = false$
Bild nicht bewegen	$beweg = false$
Gleiche Randknotenzahl auf V.-Seiten	$symmetrie = true$
Kein Tafelmenü geöffnet	$offen = false$

Tab.3. Standardinitialisierung der Parameter und Einstellungen

Zunächst eine Übersicht über das Hauptmenü sowie seine wichtigsten Bestandteile.

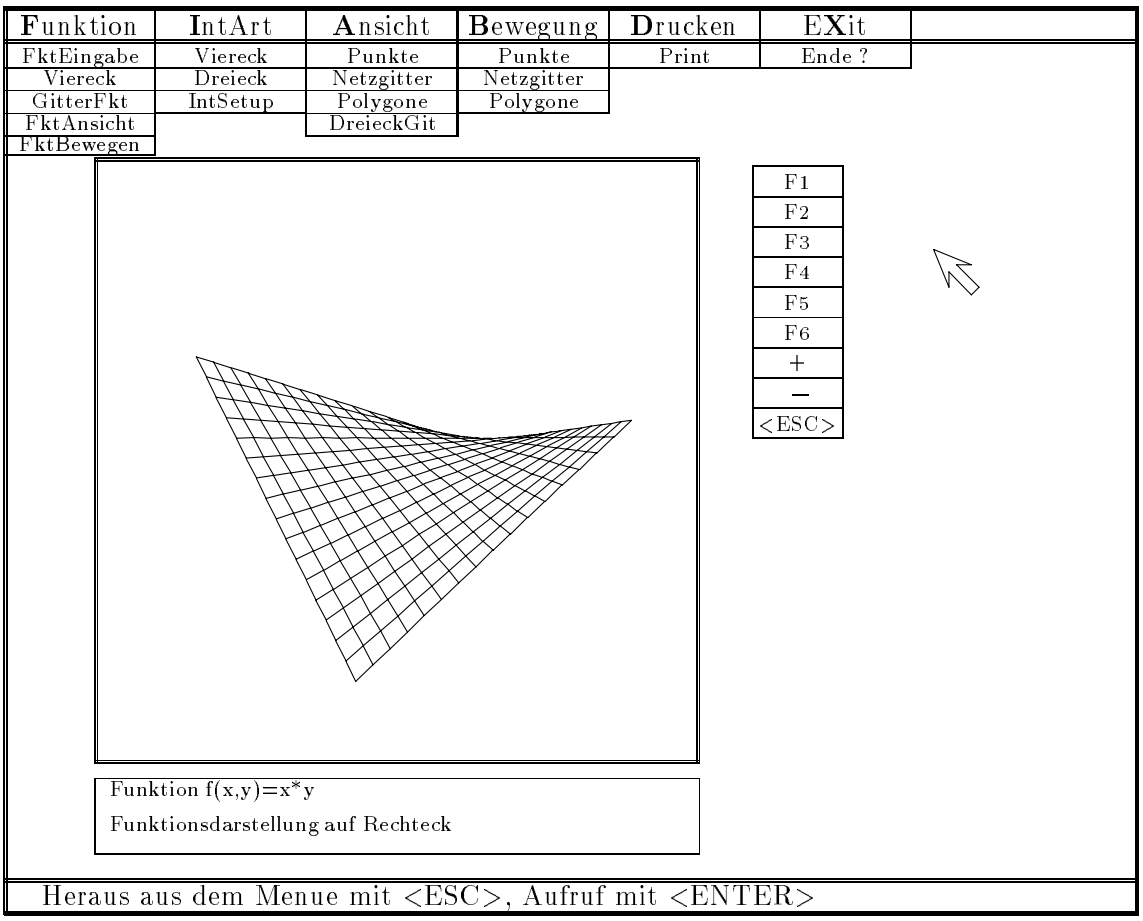


Abb.3. Menüauswahlmöglichkeiten sowie Darstellung von $f(x,y) = x\,y$ auf dem Rechteck [-1,1]*[-1,1], Gitterpunkte $idx = idy = 17$

- Menü **Funktion**
  Dieses Menü dient zunächst zur Auswahl und Betrachtung der Originalfunktion auf einem Rechteckgebiet.
  Bei Wahl des ersten Menüpunktes erscheint das Eingabefeld für die Eingabe der zu interpolierenden Funktion oder der Nummer der Formfunktion. Es ist darauf zu achten, daß die Funktion im betrachteten Bereich keine Funktionswerte hat, die zu einem Zahlenüberlauf führen, was einen sofortigen Programmabsturz zur Folge hat.
  Als zweites kann man das Rechteckgebiet verändern und die Gitterauflösung für die Anzeige einstellen.
  Drei weitere Menüpunkte dienen der Ansicht und dem Bewegen der Funktion.

– Menü **IntArt**
Wahl zwischen Rechteck oder Dreieck als Interpolationsgrundfläche.
Mittels Menüpunkt „IntSetup“ werden dann auf dem ausgewählten Grundgebiet weitere Einstellungen vorgenommen.

Interpolation auf Viereck:
1.) Eingabe der Begrenzungen des Rechtecks in der x, y-Ebene.
Es ist darauf zu achten, daß kein fast entartetes Viereck eingegeben wird, da die gesamte GK-Arithmetik nur den Typ SINGLE benutzt, und es durch Rundungsfehler zu singulären Matrizen bei der Interpolation kommen kann.
Ein eingegebenes Zeichen kann nur mit Hilfe der BACK-SPACE-Taste gelöscht werden.
2.) Eingabe der Anzahl der finiten Elemente pro Achsenrichtung.
3.) Eingabe der Randpunktanzahl je Viereckseite (Genauigkeit).
4.) Auswahl zwischen Darstellung des Interpolationspolynoms oder der Differenz zwischen Interpolationspolynom und interpolierter Funktion (Differenz nicht sinnvoll bei Formfunktionen).

Interpolation auf Dreieck:
1.) Eingabe der 3 Eckpunktkoordinaten (Hinweise wie oben).
2.) Eingabe der Anzahl der Stützstellen je Dreieckseite (Genauigkeit).
3.) Auswahl zwischen Darstellung des Interpolationspolynoms oder der Differenz zwischen diesem und interpolierter Funktion.

– Menü **Ansicht**
Rechteckgitternetz für die Darstellung = $idx * idy$ Punkte.
Hier kann zwischen Ansicht von Punkten, Drahtgitter (verdeckte Linien werden gezeichnet) und Polygonen (farbiges Gebirge, verdeckte Flächen werden nicht gezeichnet) gewählt werden.
Bei Punktansicht werden nur Punkte der Funktion an den Stützstellen dargestellt. Diese Punkte werden bei Netzgitteransicht durch Linienelemente verbunden. In der Polygondarstellung sind die in der Netzgitteransicht entstehende Segmente der Funktionsfläche farbig ausgefüllt.
Für den Dreieckfall „DreieckGit“ kann die Netzansicht auf dem Viereck auf das Dreieck abgeschnitten werden, eine Bewegung dieser ist nicht vorgesehen.
Wird ein Menüpunkt gewählt und bestätigt, wird die momentan gültige Funktion oder Formfunktion unter den momentan gültigen Interpolationsbedingungen im gewünschten Modus dargestellt.

– Menü **Bewegung**
Die Menüpunkte entsprechen bis auf den Dreieckfall denen im Ansicht-Menü. Nach dem Aufrufen eines Menüpunktes wird die Funktion im gewählten Modus dargestellt und kann über die Buttomleiste bzw. die entsprechenden Tasten wie folgt bewegt werden:

F1, F2 : Drehung um die x-Achse, Drehwinkelschritt $= \pi/30$
F3, F4 : Drehung um die y-Achse
F4, F5 : Drehung um die z-Achse
$+$, $-$: stärkere bzw. schwächere Perspektive

Durch <ESC> oder rechten Mausklick kann dieser Modus wieder verlassen werden, wobei die im Ansicht-Menü gewählte Darstellung hergestellt wird.

- Menü **Drucken**
  Druck des im Anzeigefenster sichtbaren Funktionsgraphen mit kurzem Titel. Am meisten ist bei der Ausgabe der Netzgitterdarstellung zu erkennen. Der Drucker wird bei Auswahl von „Print“ auf Bereitschaft getestet.
- Menü **Exit**
  Bestätigen mit <ENTER> bzw. ein Mausklick auf „Ende ?“ führt zum Progammende, <ESC> bzw. Aufruf eines anderen Menüs zum Verweilen im Programm.
- Weitere Optionen

  Damit sind Eingriffe in den Quelltext des Programms und zwar in die Prozedur *zeigen* verbunden. Die Stellen sind sichtbar mit „Teil I, II, III“ gekenzeichnet.

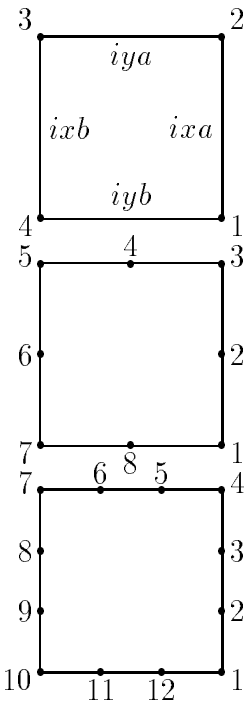
  1.) Es besteht für das Rechteck die Möglichkeit, bei Interpolation über mehrere finite Elemente (max. 4*4), für jedes Element eine andere Randpunktanzahl anzugeben. Hierzu sind entsprechende Festlegungen im Feld *genauigkeit* zu treffen. Dies ist nur durch eine Änderung im Quellcode bei Teil I möglich.
  Für ein kleines Rechteckelement werden dann die Größen ixa, iya, ixb, iya (siehe Abb.4) in Abhängigkeit von der Genauigkeit der jeweiligen (max. 4) Nachbarelemente neu berechnet ($ni = max(ixa, ixb, iya, iyb) \leq genauigkeit$) und die Übergänge bei der Interpolation berücksichtigt.

  2.) Für ein Rechteck ohne Unterteilung mit *genauigkeit* kann man ebenfalls im Nachhinein die Anzahlen ixa, iya, ixb, iya der Randknotenpunkte auf seinen Seiten neu definieren, wobei $max(ixa, ixb, iya, iyb) \leq genauigkeit$ gelten muß. Dies passiert an der Stelle Teil II.

  3.) Man kann auch weitere „Formfunktionen“ oder nicht analytisch gegebene Funktionen bearbeiten (siehe Abb.7), und das gemeinsam mit 2.). Dazu sind die einzelnen Stützstellen-z-Werte in das Feld *pkt* bei Teil III einzufügen. Vorherige Berechnungen von $pkt[*].z$ werden überschrieben.

  Es ist darauf zu achten, daß bei der folgenden Programmabarbeitung, nur die den Eintragungen entsprechenden Einstellungen zu benutzen sind, da sonst nicht das gewünschte Ergebnis erzielt wird.

- Gebietscharakteristik und Interpolationsansätze (vgl. [7])
  Alle Stützstellen der Grundgebiete sind bis auf einen Fall äquidistante Randknoten. Die „Genauigkeit“ der Polynomansätze ist 2, 3, 4 .
  Damit sind folgende Zuordnungen verbunden.



Genauigkeit = 2

Vierknotenrechtelement, bilinearer Ansatz

$p_n(x,y) = c_1 + c_2x + c_3y + c_4xy$

Genauigkeit = 3

Achtknotenrechteckelement, quadratischer Ansatz
der Serendipity-Klasse, Grad der Vollständigkeit=2

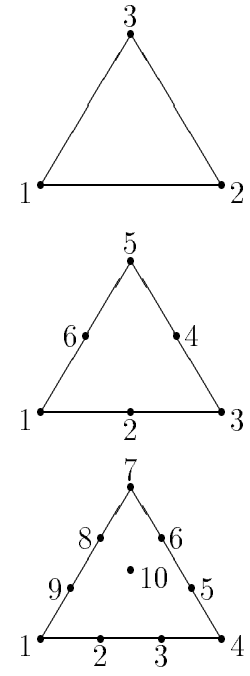
$p_n(x,y) = c_1+c_2x+c_3y+c_4x^2+c_5xy+c_6y^2+c_7x^2y+c_8xy^2$

Genauigkeit = 4

Zwölfknotenrechteckelement, Grad der Vollständigkeit=3

$$p_n(x,y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2 + c_7x^3 + c_8x^2y + c_9xy^2 + c_{10}y^3 + c_{11}x^3y + c_{12}xy^3$$

Abb.4. Viereckelemente mit Knotennumerierung und Polynomansätzen



Genauigkeit = 2

Dreiknotendreieckelement, linearer Ansatz

$p_n(x,y) = c_1 + c_2x + c_3y$

Genauigkeit = 3

Sechsknotendreieckelement, quadratischer Ansatz
Grad der Vollständigkeit=2

$p_n(x,y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2$

Genauigkeit = 4

Zehnknotendreieckelement, Grad der Vollständigkeit=3

$$p_n(x,y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2 + c_7x^3 + c_8x^2y + c_9xy^2 + c_{10}y^3$$

Abb.5. Dreieckelemente mit Knotennumerierung und Polynomansätzen

Ein Dreieck wird automatisch eingebettet in ein kleinstes achsenparalleles Rechteck. Diese Maßnahme führt erstens zur Darstellung von Funktionsgraphen über einem Viereck, und zweitens wird damit die Drehung und Streckung vereinfacht. Nachteilig ist, daß dann Funktionswerte an Argumenten außerhalb des Dreieckgebietes gemäß Gitterauflösung starke Abweichungen haben können und mit der Skalierung des Plots die Anschaulichkeit gemindert wird.

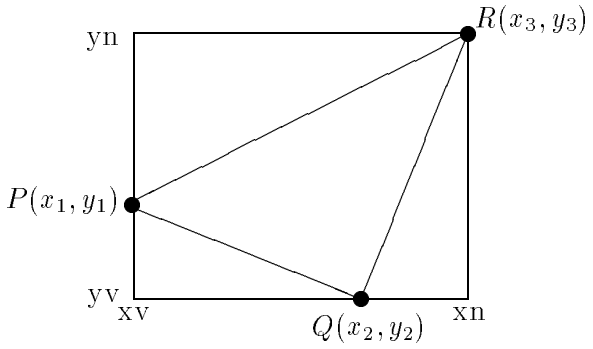


Abb.6. Dreieckeinbettung

- Funktionswahl
  Diese ist analog zu der im Programm **PLOT3D32.PAS**.
  Im Eingabefenster ist entweder die Funktion $f(x, y)$ zu definieren mit den angebotenen Möglichkeiten, oder die Eingabe spezieller Symbole (Zahlen) wird interpretiert als Auswahlnummer für eine Formfunktion, die somit die Referenz zur Interpolation liefert. Der Originalfunktion als solche wird auf dem Gebiet ein konstanter Wert zugeordnet (meist 1).
  Funktionsnummer k heißt, daß am Knoten mit der Nummer k der Funktionswert 1 ist ($pkt[k].z := 1$), die übrigen sind 0.
  Entsprechend der Anzahl der Randknoten des Elements, d.h. der Genauigkeit, sind folgende Funktionsnummern relevant (vergl. Abb. 4).

Gen.	Gebiet	
	Viereck	Dreieck
2	1, 2, 3, 4	1, 2, 3
3	1, 2, ... , 8	1, 2, ... , 6
4	1, 2, ... , 12	1, 2, ... , 10

Tab.4. Numerierung der Formfunktionen

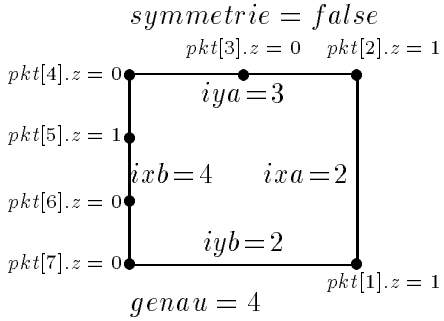


Abb.7. Rechteck mit Extra-Formf.

Bei Viereckunterteilung und Formfunktion sind letztere sowie die Genauigkeit für alle Elemente einheitlich.

- 3D-Darstellung und Interpolation
  Wir illustrieren das Ergebnis der Interpolation einer Funktion $f(x, y)$ auf einer Rechteckunterteilung mit bilinearen Ansatz.
  Folgende Einstellungen wurden für die anschließenden Plots verwendet.

Eingabe der Einstellungen fuer die Interpolation ueber dem Gebiet :	Rechteck
Intervallgrenzen :	xv = -1.00
	xn = 2.00
	yv = -1.00
	yn = 2.00
Finite Elemente :	ibx = 4
	iby = 4
Randpunkte :	pkt = 2
Differenzdarst. :	dif = N
	<OK> <ESC>

Tab.5. Menü „IntSetup“ zur Parametereinstellung

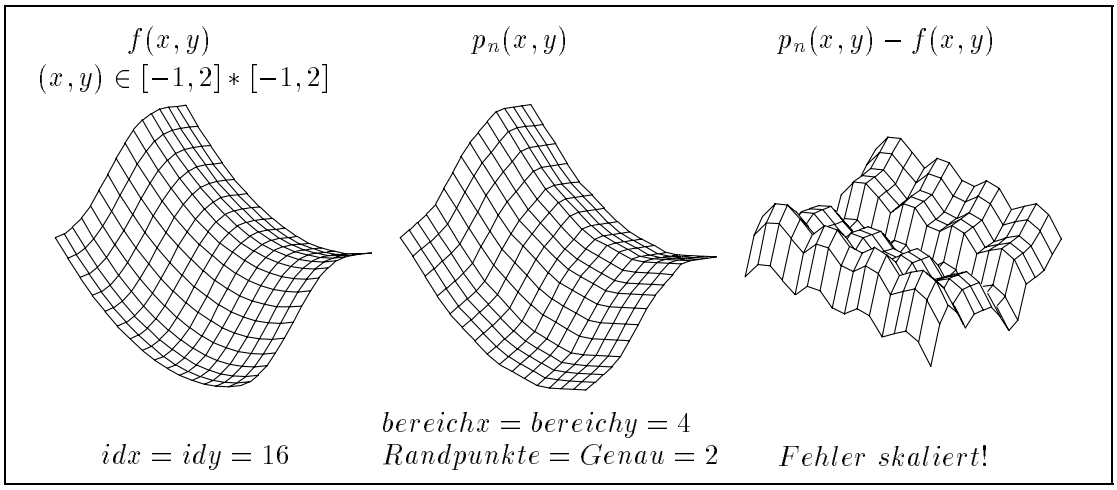


Abb.8. Interpolation und Funktionsgraphen zu $f(x, y) = 0.5\,(x^2 + cox(2y))$

Die Datenfiles dazu wurden durch das Programm erzeugt und mittels GNUPLOT weiterverarbeitet. Dabei ist eine Ausdünnung der Daten durch GNUPLOT möglich, jedoch eventuell verbunden mit der Gefahr des Verlustes an Informationen und Anschaulichkeit.

- Formfunktionen

  Zunächst geben wir auf zwei Standardgebieten die Formfunktionen an. Bezüglich des Dreiecks bemerke man, daß, obwohl das Datenfile des Programms nur x, y-Werte aus dem Dreieck enthält, GNUPLOT automatisch eine Erweiterung auf das Viereck [0,1]*[0,1] vornimmt und zwar wegen der damit verbundenen Vereinfachung des Drehmechanismus für den Graphen.

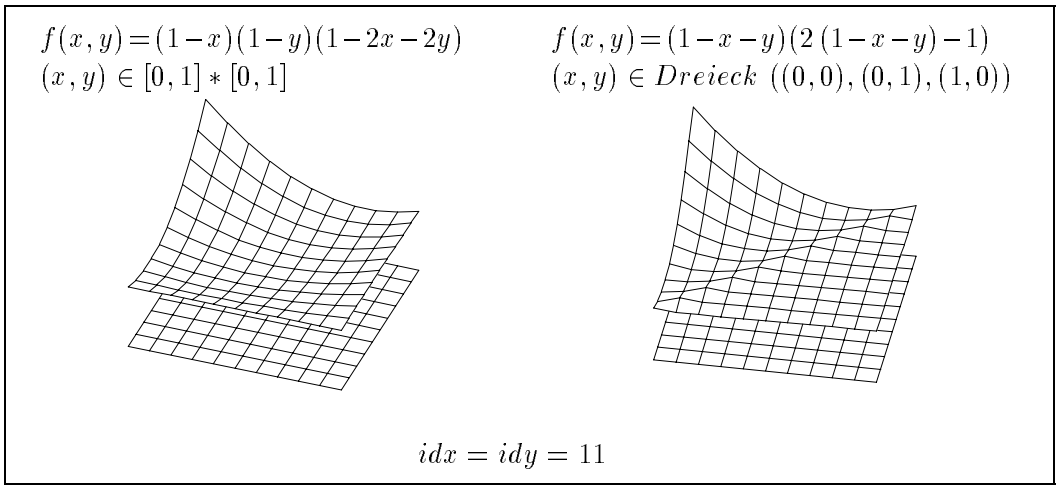


Abb.9. 1-0-Formfunktionen Nr. 1 der Genauigkeit 3

Das Rechteck kann noch einmal in $r = bereichx * bereichy$ kleinere Rechteckelemente unterteilt werden.

3	4
1	2

5	6
3	4
1	2

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

Abb.10. Verschiedene Unterteilungen mit Elementnumerierung.

Dabei werden die lokalen Genauigkeiten und Knotenpunktzahlen auf den kleinen Rechteckelementen als einheitlich angenommen. Andere Einstellungen könnten zwar im Quelltext vorgenommen werden, würden aber den Sachverhalt verkomplizieren.
Die Wahl einer Formfunktion bei Gebietsunterteilung führt dazu, daß auf jedem Element dieselbe verwendet wird und somit ein lokaler Funktionsverlauf (Gebirge) sich r Mal wiederholt. Genau genommen haben wir damit die Interpolation einer unstetigen Funktion, und das Programm müßte eigentlich abbrechen (Nulldivision). Nur durch eine nichtexakte Zahlendarstellung im Rechner kann diese Hürde eventuell übersprungen werden. Jedoch entstehen an den Elementrändern sehr steile Funktionsverläufe, die ihrerseit wiederum Einfluß auf die Skalierung und die Größe des Plots haben. Erkennbar wird dieser Effekt, falls die Stützstellen der Interpolation (Randknoten) eine Teilmenge derer der

Gitterauflösung oder umgekehrt sind.
Bricht die Berechnung nicht ab und stimmen die Interpolationsstellen mit denen der Gitterauflösung nicht überein, fallen die nebeneinander liegenden Gebirge unterschiedlich hoch aus.

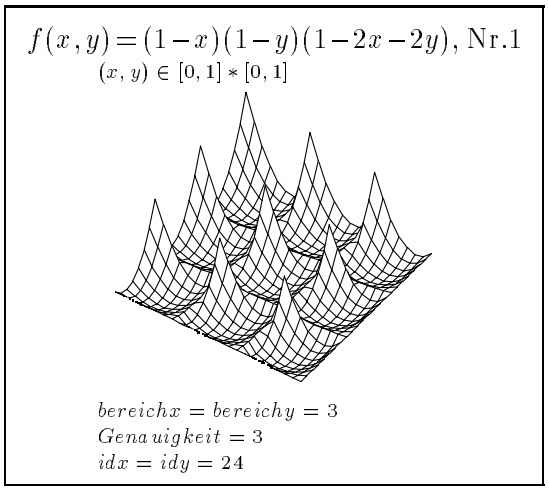


Abb.11. Interpolation bei Rechteckunterteilung

- Programmkomponenten
  - Bild bewegen (Animation)
    Die Graphikkarte VGAHi erlaubt es, zwei Bildschirmseiten zu bearbeiten (Nummer 0, 1).
    Das Umschalten auf die aktive BS-Seite erfolgt mit $SetActivePage(Nr)$. Welche Seite aber auf dem Bildschirm erscheint, wird mittels $SetViewPage(Nr)$ festgelegt.
    Die Kombination erlaubt bei Anzeige von Seite 0 die Vorbereitung eines Bildes auf der aktiven aber nicht sichtbaren Seite 1. Ist dies erfolgt, so wird wieder die Seite 0 zur aktiven und mit einem schnellen Kopierbefehl unter Nutzung von Portadressen und Speicheradressierung ($mem[seg:ofs]$) wird das Bild auf Seite 0 in den Anzeigebereich des Bildschirms bewegt. Wegen des schnellen Ablaufs dieser Befehle erscheint dies im Menü „Bewegung“ wie eine kontinuierliche Bildveränderung.
    Der Zugriff auf das Zeichen an der Stelle $(x,y), 1 \leq x \leq 80, 1 \leq y \leq 25$, des Bildschirms erfolgt über den mem-Befehl. Die beiden Adreßteile sind das Segment seg = \$B800 (Videosegmentadresse) für den BS-Speicher sowie das Offset gemäß $ofs = 2*(x-1+80*(y-1))$ (Zeichen und sein Attribut belegen zwei aufeinanderfolgende Bytes).
    Die BS-Seiten 0 und 1 befinden sich ab Segmentadresse \$A000 : \$0000 bzw. \$A000 : \$9600. Der jeweilige Zugriff auf die 640*480 Pixel dieser Seiten geschieht mittels der entsprechenden Offsetadresse. Die Seite wird dabei betrachtet wie eine Folge von 480 Zeilen zu je 80 Byte (1 Byte umfaßt 8 Pixel). Die Situation kann man vereinfacht schematisch darstellen.

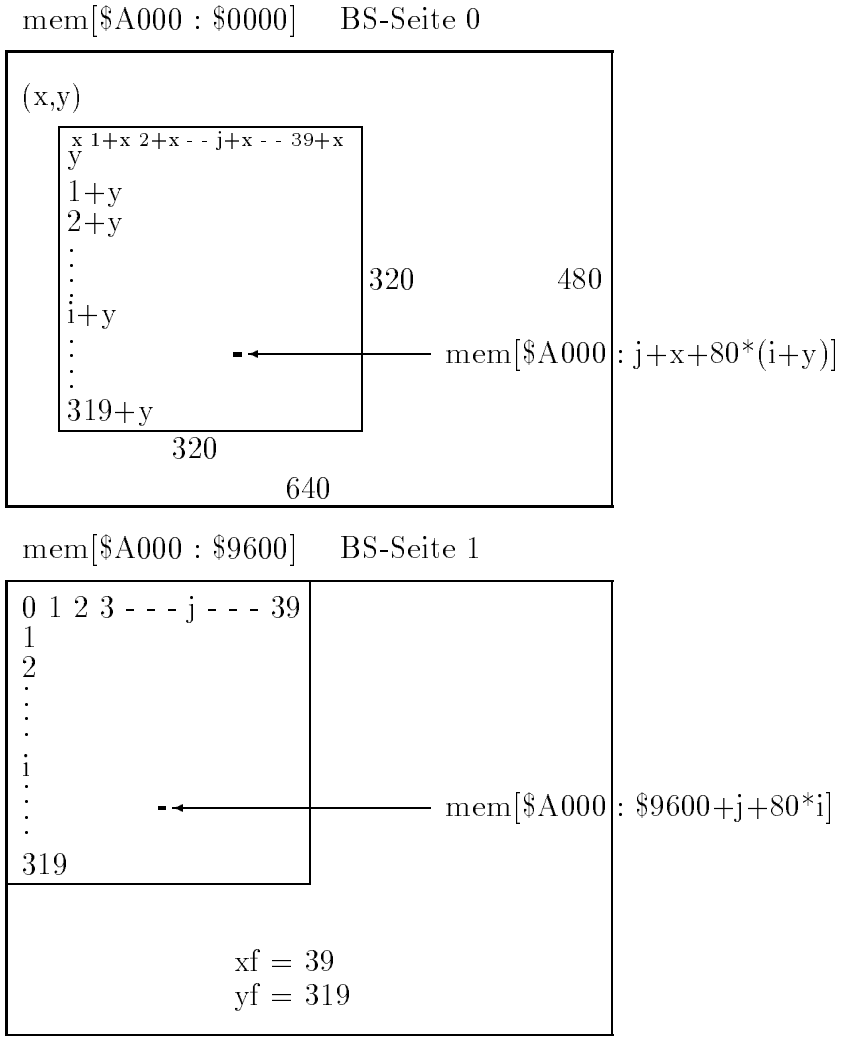


Abb. 12. Schema der Verknüpfung der Bildschirmseiten.

Die programmtechnische Umsetzung hat folgende Form.

```
{ Dimensionen des Anzeigefensters }
const xf = 39;
      yf = 319;

procedure page_copy(x,y : integer);
const seg : Word = $A000;
      ofs : Word = $9600;  { =38400 }
var   i,j : integer;
begin
  Port[$03CE]:=$05;
  Port[$03CF]:=(Port[$03CF] and $FC)+1;
  for i:=0 to yf do
    for j:=0 to xf do
      mem[seg:j+x+80*(i+y)]:=mem[seg:ofs+j+80*i];
  line(getmaxx,getmaxy,getmaxx,getmaxy);
end;
```

```
procedure draw1;          { Punkte zeichnen }
var i,j : byte;
begin
  setactivepage(1);
  { Pixeleintraege in aktive Seite }
  setfillstyle(1,1);
  bar(0,0,(xf+1)*8,yf);
  for i:=1 to idx do
    for j:=1 to idy do
      putpixel(scr[i,j][1],scr[i,j][2],15);
  setactivepage(0);
  page_copy(xwin,ywin);   { xwin=48, ywin=70 }
end;
```

- Maus für Text bzw. Graphik-Bildschirm ein/aus mittels Software-Interrupt $33=33h=51

```
unit MAUS;

interface
var Fehler : Integer;
    xm,ym  : Integer;   { Mausposition fuer Abfrage }
    Button : Byte;      { Mausschalter 1,2 }

procedure InitMouse;
procedure ShowMouse;
procedure HideMouse;
function  StatusMouse : Byte;

implementation
uses DOS;
var  R,G : Registers;

procedure InitMouse;
begin
  R.AX:=0;
  Intr($33,R);
end;
procedure ShowMouse;
begin
  R.AX:=1;
  Intr($33,R);
end;
procedure HideMouse;
begin
  R.AX:=2;
  Intr($33,R);
end;
function StatusMouse : Byte;
begin
   Button:=0;
   G.AX:=3;
   Intr($33,G);
   Button:=G.BX;
   xm:=G.CX;
   ym:=G.DX;
end;
```

2.3 Was leisten Computer und Software?

An solchen Beispielen zeigt sich, wie mit Computereinsatz und Software die Anschauung und der Erkenntnisprozeß unterstützt werden können. Der wirklich kreative Anteil am Problemlösungsprozeß bleibt beim Menschen. Der Phantasie sind aber keine Grenzen gesetzt. Natürlich ist es zumeist nicht einfach, komplizierte Sachverhalte methodisch geschickt und didaktisch wirksam für Präsentationen vor dem betreffenden Hörerkreis aufzubereiten. Aber dazu gibt es schon zahlreiche Erfahrungen und Modelle, neue und erfolgversprechende Wege werden ständig gegangen.

Literatur

[1] DÖRFLER, W.: *Der Computer als kognitives Werkzeug und kognitives Medium.* In : Computer-Mensch-Mathematik, Schriftenreihe Didaktik der Mathematik Universität Klagenfurt, Band 21, 1994.

[2] NEUNDORF, W.: *Kondition eines Problems und angepaßte Lösungsmethoden.* Preprint No. M 9/95, TUIlmenau FMN IfMath April 1995.

[3] RUBENKING, N.: *Turbo Pascal 6.0. Profitechniken und Tools.* te-wi Verlag GmbH München 1992.

[4] BEISECKER, M.-A.; BRICKWEDE, P.(Hrsg.): *Turbo Pascal Power Tools.* SYBEX Düsseldorf 1990.

[5] SZCZEPKOWICZ, J.: *Turbo Pascal 5.0. z przykladami konstrukcji oprogramowania podstawowego.* Wydawnictwa Naukowo-Techniczne Warszawa 1990.

[6] KIELBASINSKI, A.; SCHWETLICK, H.: *Numerische lineare Algebra.* Mathematik für Naturwissenschaft und Technik Band 18, DVW, Berlin 1988.

[7] SCHWARZ, H.R.: *Methode der finiten Elemente.* LAMM Band 47. Teubner Studienbücher Mathematik. B.G.Teubner Stuttgart 1991.

[8] KOTZ,D.: *LaTeX and the GNUPLOT Plotting Program.* GNUPLOT LaTeX Tutorial Version 3.0, 1991.

[9] FOLEY, J.D., van DAM, A. u.a.: *Computer Graphics.* Addison-Wesley Publ. Company Bonn 1990.

[10] ENCARNAÇÀO, J., STRASSER, W.: *Computer Graphics.* Gerätetechnik, Programmierung und Anwendung graphischer Systeme. R.Oldenbourg Verlag München 1988.

[11] BARTH, R., BEIER,E. ,PAHNKE, B.: *Graphikprogrammierung mit OpenGL.* Reihe Praktische Informatik. Addison-Wesley Publ. Company Bonn 1996. ISBN 3-89319-975-6.

Anschrift:

Stud. cand. Thomas Lutter, Dr. Werner Neundorf
Technische Universität Ilmenau Institut für Mathematik
PF 10 0565
D - 98684 Ilmenau
e-mail : neundorf@mathematik.tu-ilmenau.de